CS-233 Theoretical Exercise 9

April 2024

1 Convolution - Computation

Let us denote an image as X. We can index a pixel x in this image as $x = X_{ijc}$ (or equivalently in a more programming-oriented format $X[i, j, c]$), where i, j, c are the indices for height (H), width (W), and channel (C), respectively.

Now let us consider a 2D convolution kernel K, with parameters $\{k \mid k = K_{ijc}\}$ and height H_K, width W_K, and number of channels C_K. We want to compute the convolution of this kernel with the input image X. Assume that we use stride s and padding p.

Question 1 [feature map size]: Give the expressions of the height, width, and number of channels of the output feature map. Assume that the stride s is carefully chosen so that the kernel convolves the image entirely, without any margin.

Solution:

$$H_F = (H - H_K + 2p)/s + 1$$
$$W_F = (W - W_K + 2p)/s + 1$$
$$C_F = 1$$

Here we assume that $(H - H_K + 2p)/s$ and $(W - W_K + 2p)/s$ are integers.

Question 2 [computation cost]: Now instead of a single kernel, we have C_{out} convolution kernels. If we just consider multiplication as one operation, how many operations are needed for the overall computation? Compare it to the number of operations we need if the output feature map is produced by a fully-connected layer instead (you can omit the bias).

Solution:

Convolution: $C_{out} H_F W_F H_K W_K C$
Fully-connected: $C_{out} H_F W_F H W C$
The number of operations is reduced by a factor of $\frac{H_K W_K}{H W}$.

Question 3 [alternative view of convolution]: Now consider a different problem: We have a black canvas of size 50×50 with two white points at positions [20, 40] and [40, 20] (index starting from 0, shown in Fig. 1). We would like to obtain two Gaussian blobs (of size 10×10) centered at the two white points, as shown in Fig. 2. Write pseudo-code to do so with only a convolution.

Solution:

This can be achieved by convolving a Gaussian image with the original canvas. The following is a working code example:

```
import numpy as np
from scipy import signal

canvas = np.zeros((50, 50))
canvas[20, 40] = 1
```

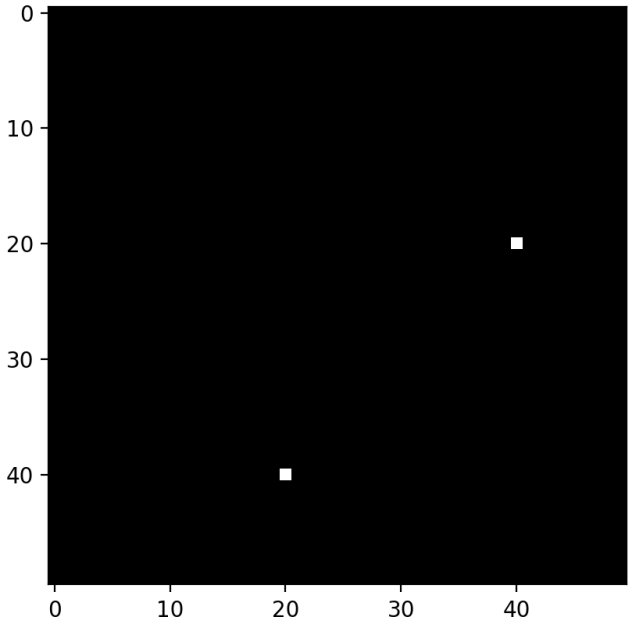

Figure 1: Black canvas with two white points

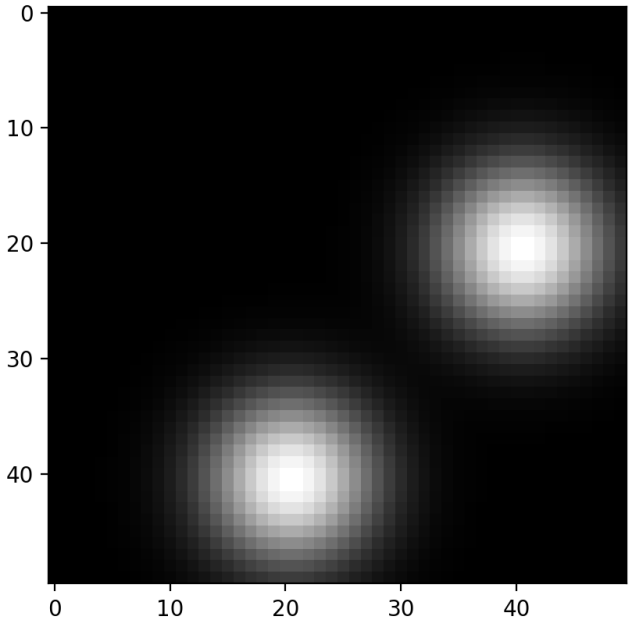

Figure 2: Desired output

```
canvas[40, 20] = 1

guassian_1d = signal.gaussian(50, std=5)
gaussian_2d = gaussian_1d.reshape(-1, 1) @ gaussian_1d.reshape(1, -1)
result_gaussian = signal.convolve2d(canvas, gaussian_2d, mode='same')
```

2 Convolution - Optimization

Question 4 [David's CNN]: David designed a CNN for dog/cat classification. He stacked multiple convolutional, average pooling, and fully-connected layers. However, he didn't use any non-linear activation functions, except for the last softmax operation. He believes that his CNN will surely outperform a simple logistic regression in terms of training loss. Explain why this is not correct.

Solution: A network without any non-linear operations is purely linear, thus a subclass (because the parameters are constrained) of logistic regression in this case. Therefore, the training loss of David's CNN will be strictly higher than or equal to a simple logistic regression, assuming proper optimization.

Now, let us go back to the basic case of convolution between an input image X and one filter K, with no padding and a stride of 1. The output feature map is denoted as F. To simplify the case even further, let us assume that the input image only has 1 channel.

To optimize the filter K, we need to compute $\frac{dL}{dK}$, which is a function (denoted as g_K) of $\frac{dL}{dF}$ and X. To perform an adversarial attack on the input X, we need to compute $\frac{dL}{dX}$, which is a function (denoted as g_X) of $\frac{dL}{dF}$ and K. Note that all these gradients are in matrix form, *e.g.,* $\frac{dL}{dK}$ is a matrix with elements $\frac{dL}{dK_{ij}}$.

Question 5 [gradient of the filter]: As mentioned above, $\frac{dL}{dK} = g_K(\frac{dL}{dF}, X)$. What is g_K (expressed in terms of $\frac{dL}{dF}$ and X)? You should be able to get a very simple expression. **Hint:** Consider a 3x3 input with a 2x2 kernel, write down the expression of the elements in the output feature map (2x2), and compute the gradient. This should give you something interesting.

Solution: $\frac{dL}{dK} = convolve(\frac{dL}{dF}, X)$, where we regard $\frac{dL}{dF}$ as the **kernel** and X as the **input**.
See these slides for an illustration.

Question 6 [gradient of the input image]: As mentioned above, $\frac{dL}{dX} = g_X(\frac{dL}{dF}, K)$. What is g_X? You should obtain a slightly more complex expression than in the last question. **Hint:** Consider a kernel, denoted as K_{rot}, corresponding to K rotated by 180 degrees.

Solution: Let $\tilde{dF}$ be the result of zero-padding $\frac{dL}{dF}$ with $(H_K - 1)$ and $(W_K - 1)$ elements on the height and width respectively.
We have: $\frac{dL}{dX} = convolve(\tilde{dF}, K_{rot})$, where we regard K_{rot} as the **kernel** and $\tilde{dF}$ as the **input**.
See these slides for an illustration.